# Разработка учебной вариации шифрования по ГОСТ 34.12-2018 «Кузнечик» на python

Оглавление



## 1.1. ГОСТ Р 34.12-2015/34.12-2018 алгоритмы блочного шифрования «Кузнечик» и «Магма»

Российский стандарт шифрования Р 34.12-2015/34 включает в себя описание алгоритмов «Кузнечик» и «Магма». «Кузнечик» – наиболее совершенный и защищённый симметричный алгоритм блочного шифрования, с размерностью блока 128 бит и ключом 256 бит, где для генерации раундовых ключей используется SP (Substitution-Permutation network) подстановочно-перестановочная сеть (сеть Фейстеля), что представляет собой разбиение данных на части и их преобразование в несколько раундов, обеспечивая высокий уровень перемешивания данных. Шифр на основе такой сети получает на вход блок и ключ и производит некоторое количество чередующихся раундов подстановки и перестановки.

Каждый раунд включает в себя линейное и нелинейное преобразование, а также операцию наложения итерационного ключа. Всего раундов 10, 9 из которых полные и 1 финальный только с наложением последнего ключа.

Итерационные ключи получаются путём выполнения определённых преобразований над изначальным 256-битным ключом. Первые два раундовых ключа получаются путём разбиения первичного ключа на две равные части $K_1, K_2$. Дальнейшие пары генерируются с помощью применения восьми итераций сети Фейстеля, где для каждой итерации используется константа, вычисляемая путём применения линейного преобразования к номеру итерации.

Для преобразований используются S-блоки, которые должны быть заполнены случайными значениями без повторений с обеспеченной нелинейностью. Эти блоки

позволяют выполнять замещение и линейное преобразование L основанное на операциях в поле Галуа GF($2^8$) по модулю для неприводимого полинома $X^8+X^7+2^6+X+1$. Значения подстановки заданы в виде массива S' = (S'(0), S'(1), …, S'(255)) для 16 байт.

***Стоит обратить внимание на порядок нумерации байтов в алгоритме!***

### 1.2. Реализация алгоритма

Необходимо написать программу, реализующую алгоритм шифрования/дешифрования «Кузнечик» на python, позволяющую зашифровать и расшифровать передаваемое сообщение соответствующим образом.

- Определение констант
  - Необходимо определить π0 и π1 S-блоки для дальнейших нелинейных преобразований. Блоки заполняются случайными значениями без повторов, соответственно полученному варианту.
  - Задать полином для умножения в поле Галуа GF($2^8$). Выбирается согласно варианту задания
- Базовые операции
  - Реализовать операцию XOR для байтовых последовательностей
  - Написать функцию умножения GF($2^8$)
- Линейное преобразование L
  - Реализовать функцию R – циклический сдвиг и XOR
  - Соединить L в виде N-кратного применения функции R
- Обратные преобразования
  - Написать обратные функции для R и L
- Генерация ключей
  - Разбить ключ на два равных блока $K_1, K_2$
  - Сгенерировать 10 раундовых ключей через итеративное применение L и S-блоков
- Шифрование блока текста
  - Провести 9 полных раундов
    - XOR с раундовым ключом
    - Применить S-блок π0
    - Выполнить L преобразование

    - Заключительный XOR с 10-м ключом
- Дешифрование блока
    - Развернуть порядок раундовых ключей
        - Применить обратную L
        - Использовать S-блок π1
        - XOR с ключом
- Обработка данных
    - Дополнение входных данных до необходимого размера
    - Разбиение на блоки соответствующего размера
    - Поочередное шифрование/дешифрование блоков
- Вернуть зашифрованное/дешифрованное сообщение

### 1.3. Задание на работу

- ***Написать программу реализующую вручную учебную вариацию алгоритма шифрования «Кузнечик» на python, со следующими минимальными требованиями:***
    - ***Реализовать генерацию значений для S-блоков в зависимости от поступающего ключа каким-либо алгоритмом, учитывая вариант задания***
    - Реализовать вручную соответствующие операции XOR для байтовых последовательностей
    - Реализовать функцию умножения в поле Галуа для заданного вариантом полинома
    - Реализовать функции для линейного преобразования и их обратные версии
    - Реализовать функцию циклического сдвига
    - Реализовать функцию генерации раундовых ключей
    - Реализовать функцию шифрование/дешифрования блоков текста в 10 раундов
    - Чтение и запись из/в текстовый файл исходного/зашифрованного сообщения (и метаданных)

### 1.4. Варианты заданий на лабораторную работу, определяются по номеру в группе

1. Полином - |**0x11B** | $x^8 + x^4 + x^3 + x + 1$|, размер блока 4 байт

2. Полином - | **0x11D** | $x^8 + x^4 + x^3 + x^2 + 1$|, размер блока 8 байт
3. Полином - | **0x12B** | $x^8 + x^5 + x^3 + x + 1$|, размер блока 16 байт
4. Полином - | **0x14D** | $x^8 + x^6 + x^3 + x^2 + 1$|, размер блока 4 байт
5. Полином - | **0x171** | $x^8 + x^6 + x^5 + x^4 + 1$|, размер блока 8 байт
6. Полином - | **0x1F5** | $x^8 + x^7 + x^6 + x^5 + x^2 + 1$|, размер блока 16 байт
7. Полином - | **0x1C3** | $x^8 + x^7 + x^6 + x + 1$|, размер блока 4 байт
8. Полином - | **0x187** | $x^8 + x^7 + x^2 + x + 1$|, размер блока 8 байт
9. Полином - | **0x1A9** | $x^8 + x^7 + x^5 + x^3 + 1$|, размер блока 16 байт
10. Полином - |**0x11B** | $x^8 + x^4 + x^3 + x + 1$|, размер блока 8 байт
11. Полином - | **0x11D** | $x^8 + x^4 + x^3 + x^2 + 1$|, размер блока 16 байт
12. Полином - | **0x12B** | $x^8 + x^5 + x^3 + x + 1$|, размер блока 4 байта
13. Полином - | **0x14D** | $x^8 + x^6 + x^3 + x^2 + 1$|, размер блока 8 байт
14. Полином - | **0x171** | $x^8 + x^6 + x^5 + x^4 + 1$|, размер блока 16 байт
15. Полином - | **0x1F5** | $x^8 + x^7 + x^6 + x^5 + x^2 + 1$|, размер блока 4 байт
16. Полином - | **0x1C3** | $x^8 + x^7 + x^6 + x + 1$|, размер блока 8 байт
17. Полином - | **0x187** | $x^8 + x^7 + x^2 + x + 1$|, размер блока 16 байт
18. Полином - | **0x1A9** | $x^8 + x^7 + x^5 + x^3 + 1$|, размер блока 4 байт
19. Полином - |**0x11B** | $x^8 + x^4 + x^3 + x + 1$|, размер блока 16 байт
20. Полином - | **0x11D** | $x^8 + x^4 + x^3 + x^2 + 1$|, размер блока 4 байт
21. Полином - | **0x12B** | $x^8 + x^5 + x^3 + x + 1$|, размер блока 8 байта
22. Полином - | **0x14D** | $x^8 + x^6 + x^3 + x^2 + 1$|, размер блока 16 байт
23. Полином - | **0x171** | $x^8 + x^6 + x^5 + x^4 + 1$|, размер блока 4 байт
24. Полином - | **0x1F5** | $x^8 + x^7 + x^6 + x^5 + x^2 + 1$|, размер блока 8 байт
25. Полином - | **0x1C3** | $x^8 + x^7 + x^6 + x + 1$|, размер блока 16 байт
26. Полином - | **0x187** | $x^8 + x^7 + x^2 + x + 1$|, размер блока 4 байт
27. Полином - | **0x1A9** | $x^8 + x^7 + x^5 + x^3 + 1$|, размер блока 8 байт

### 1.5. Загрузка результата выполнения лабораторной работы

В качестве результата выполнения лабораторной работы необходимо загрузить в Moodle следующие файлы:

- Код реализованной программы на python
- Текстовый файл с исходным и зашифрованным текстом, а также другими соответствующими варианту данными, сгенерированными S-блоками